

Understanding the Impact of Quantum Noise on Quantum Programs

Zhonghao Pan

*State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing 210023, China
zhonghaopan@smail.nju.edu.cn*

Yang Feng

*State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing 210023, China
fengyang@nju.edu.cn*

Zhiyuan Li

*Beijing Academy of
Quantum Information Sciences
Beijing, China
lizhiyuan@baqis.ac.cn*

Yunxin Liu

*Institute for AI Industry Research (AIR)
Tsinghua University
Beijing, China
liuyunxin@air.tsinghua.edu.cn*

Yuanchun Li

*Institute for AI Industry Research (AIR)
Tsinghua University
Beijing, China
liyuanchn@air.tsinghua.edu.cn*

Abstract—Quantum computing is expected to introduce the next era of computing speed and power, and its software - quantum program is gaining increasing research interest in the software engineering community. A significant characteristic of quantum computing is the existence of noise. Unlike classical computers where the output of a program is usually deterministic, the execution of a quantum program may be affected by quantum noise. Such a difference may cause difficulties or misunderstandings for developers shifting from classical programming to quantum programming. To understand the impact of quantum noise on quantum programs and its implications for software developers, we conduct a series of studies with real-world quantum programs and quantum computing environments. Specifically, we first measure and analyze the noise in a real quantum computer by testing it with a basic quantum program. We find that a non-neglectable amount of quantum noise generally exists in real quantum computers. Then we investigate the robustness of quantum programs against different quantum noises by testing 18 real-world quantum programs and 50,000 randomly generated quantum circuits in simulated and real environments. We observe that quantum noise can significantly influence the correctness of quantum programs, and different quantum circuit structures show diverse sensitivity patterns under the same noise. Based on the observations, we build a machine learning model to predict the fidelity of a quantum program under certain quantum noise. The model achieves a small average fidelity prediction error, meaning the impact of noise can be precisely estimated statistically.

Index Terms—quantum computing, quantum noise, quantum software engineering, robustness, empirical study

I. INTRODUCTION

Quantum computing is a promising technology that enables exponentially faster and more efficient calculations as opposed to classical computers. Quantum algorithms can achieve significantly better performance on certain problems such as prime factorization and random number generation. In recent years, many research institutes and corporations including Google, Intel, Microsoft, and IBM have put much effort into developing quantum computers and quantum programming languages (*e.g.*, Qiskit [1], Q# [2], Cirq [3], etc). Some of

them have hosted quantum computers in the cloud, allowing developers to test their own quantum programs.

A quantum program is a combination of quantum parts (namely quantum circuits) and classical parts [4]. A quantum circuit that is executed on quantum computers and returns outputs through measurement gates is a collection of quantum gates interconnected by wires. The classical part is responsible for adjusting the logic of constructing quantum circuits and handling their results. Developers can design quantum programs to implement quantum algorithms including Shor's algorithm [5], Grover's algorithm [6], and Hidden Shift algorithm [7].

A major characteristic of quantum computers is the existence of noise, which will lead to state decoherence and information loss [8]. Quantum noise is originated from multiple sources, including temperature, charge effect, quantum gate accuracy, etc. Because of the complexity of the sources, physicists haven't found rigorous ways to explain and eliminate the noise so far. This is a major challenge that limits the development of quantum computing [9].

Quantum noise may lead to unexpected errors for programs running on quantum computers, *i.e.*, programs that are correct in principle may produce failures when being executed on real devices [9]. Such a phenomenon seldom appears in classical software since classical computers are mostly noise-free and classical instruction sets are deterministic. Thus, understanding the quantum noise is a key step when software developers shift from classical programming to quantum programming.

However, knowledge about quantum noise is deficient in the software engineering community. The extent and characteristics of the quantum noise in real quantum computers are largely unknown, and it is also unclear how a quantum program is affected by the noise [10]. Although there are already dozens of studies on quantum programming and quantum software engineering [4], [11], the existence of noise is usually neglected.

To this end, we conduct a series of studies to deepen the understanding of quantum noise in quantum computing.

Specifically, we aim to answer three research questions:

- **RQ1:** What are the characteristics of noise presented in real quantum computers?
- **RQ2:** How does the noise influence the correctness of quantum programs?
- **RQ3:** Whether and how the impact of quantum noise can be modeled?

We attempt to answer these research questions by studying real quantum computers and quantum programs. Specifically, we mainly consider two computing environments, including a quantum computing cloud platform [12] and a quantum simulator [3]. We also collect 18 real quantum programs from GitHub, including the implementations of many classical quantum algorithms. Additionally, we generate more than 50,000 quantum circuits for larger-scale analyses.

Based on the collected and simulated quantum programs and computing environments, we first investigate the characteristics of real quantum noise by running a benchmark circuit on the quantum computing cloud platform. Then we analyze the robustness of quantum programs against noise by testing the programs with different types and levels of quantum noise. Finally, we build a graph neural network to model the impact of quantum noise on quantum programs. Our findings include:

- 1) A non-negligible scale of quantum noise is generally present in tested quantum computers, and the strength of the noise varies over time.
- 2) The correctness of quantum programs may be significantly influenced by quantum noise. For example, when the bit-flip noise probability is larger than 0.2, the average accuracy of 14 quantum programs is reduced to 17.7%. We also notice that different programs (even the programs with equivalent functionality) have different levels of robustness in the same noise computing environment.
- 3) Our machine learning model can predict the fidelity of a quantum circuit with a small average error of about 0.05, meaning that the impact of quantum noise on quantum programs can be numerically estimated.

We believe our findings can help quantum programmers better understand quantum noise and assess the robustness of their programs in advance to optimize them accordingly.

This paper makes the following contributions:

- To the best of our knowledge, this paper is the first study of quantum noise from the software engineering perspective.
- Our study produces several interesting findings, including the noise characteristics in real quantum computers and the robustness of quantum programs against noise.
- We build a graph neural network to model the quantum noise in a given environment, which can estimate the impact of quantum noise on a quantum program numerically.
- We release our dataset and highlight several important takeaway messages for quantum software developers.

II. BACKGROUND AND RELATED WORK

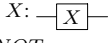
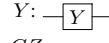
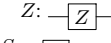
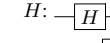
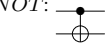
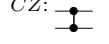
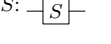
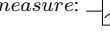
A. Quantum Bit and Quantum Gate

The smallest unit in quantum computing is a **quantum bit (or qubit)**, whose value can be $|0\rangle$, $|1\rangle$, or a superposition of

TABLE I
COMMON QUANTUM GATES AND THEIR OPERATION MATRICES.

Gate	Matrix	Gate	Matrix	Gate	Matrix
T	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$	X	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	X2M	$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & i \\ i & 1 \end{bmatrix}$
TD	$\begin{bmatrix} 1 & 0 \\ 0 & e^{-i\pi/4} \end{bmatrix}$	Y	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$	Y2M	$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$
S	$e^{-i\pi/4} \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}$	Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	X2P	$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -i \\ -i & 1 \end{bmatrix}$
SD	$e^{i\pi/4} \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}$	H	$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	Y2P	$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$
CZ	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	CNOT	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$		

TABLE II
GRAPHICAL REPRESENTATION OF COMMON QUANTUM GATES.

X: 	Y: 	Z: 	H: 
CNOT: 	CZ: 	S: 	measure: 

both. In general, a qubit is mathematically represented as a **quantum state** of the form:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \alpha, \beta \in \mathbf{C}, |\alpha|^2 + |\beta|^2 = 1$$

where $|\alpha|^2$ and $|\beta|^2$ are the probabilities of the quantum state being $|0\rangle$ and $|1\rangle$ respectively. When we measure the quantum state, it randomly collapses into one of the basic states $|0\rangle$ or $|1\rangle$ according to its probabilities.

The quantum state of multiple involved qubits can be represented as their direct product. For example, $|\psi_1\psi_2\rangle$ represents the state of two qubits, written by:

$$|\psi_1\psi_2\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix} \otimes \begin{bmatrix} \alpha_2 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \alpha_1\alpha_2 \\ \alpha_1\beta_2 \\ \beta_1\alpha_2 \\ \beta_1\beta_2 \end{bmatrix} \quad (1)$$

The quantum states may change after going through **quantum gates**. A quantum gate that acts on n qubits is represented by a $2^n \times 2^n$ unitary operation matrix. Table I shows some common quantum gates and their operation matrices, and some of their graphical representations can be found in Table II:

The output of a quantum gate is the result of multiplying the matrix by the input quantum state. Taking the X gate, for example, its resulting quantum state after operating on $|\psi\rangle$ is:

$$X|\psi\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

B. Quantum Programming

In classical computers, a program is consist of variables, statements, and functions. However, in a quantum program, besides the classical part, the main logic is implemented by **quantum circuits**, which are constructed with multiple connected quantum gates. An example quantum circuit is shown in Figure 1, where the single lines denote the channel of qubits, and doubled lines transport classical bits (0 and 1). Each qubit corresponds to a single line channel with a sequence of quantum

gates on it indicating the operations on each qubit designed by programmers on purpose. In a quantum circuit, a set of longitudinal gates is called a moment. We define the number of moments in a circuit as its depth. After measurement, the qubits are collapsed to classical bits based on the probabilities described in their final states, and then we get the result with a sequence of numbers like "000" or "111", but not the quantum states. Thus a circuit usually needs to be executed a lot of times to obtain the precise output.

```

qubit = cirq.GridQubit(0,0) # Pick a Qubit
circuit = cirq.Circuit([    # Create a Circuit
    cirq.X(qubit),          # Add a NOT Gate
    cirq.measure(qubit)     # Add a measurement Gate
])
s = cirq.Simulator()        # Create a simulator
r = s.run(circuit, repetitions=10) # Run the circuit
print("Results: ", r)      # Print the results

```

Code 1. An example of quantum program in Cirq.

Developers use quantum programming languages to design quantum circuits and the classical logic for operating (*i.e.*, constructing, invoking, and processing results of the circuits [13]). Code 1 shows a Cirq [3] program.

A quantum program can then be executed in quantum computing environments. There are mainly two types of them available for most developers today:

- **Quantum cloud platforms** are online services hosted by quantum computer owners. Developers can submit a quantum circuit to the cloud platform, and the quantum computer linked to it will execute the quantum circuit and return the result to programmers. Unfortunately, due to the expensive cost and immature technology, publicly-accessible quantum computing platforms are rare.
- **Quantum simulators** are classical software developed for the simulation of quantum computers. The simulator is based on the computational implementation of the effect of quantum operations, which is time-consuming and may not exactly imitate real quantum computers since the environmental noise cannot be perfectly replicated.

C. Quantum Noise

A major characteristic of quantum computing is the existence of quantum noise, which may prominently affect the quantum states in a quantum computer by causing decoherence. Therefore, unlike classical programs that can return deterministic outputs, quantum programs running on quantum computers may not produce the theoretical results expected by the developers. There are various causes of quantum noise [14], including:

- **Interaction with the environment.** Qubits need to be perfectly isolated from the environment to maintain their state, or the quantum state will change consistently.
- **Interaction with other qubits.** Qubits also influence each other. Quantum interactions are necessary for quantum programming and the inter-qubit influence is unavoidable.
- **Imperfect operations.** The quantum gates that act on quantum states cannot be perfectly implemented. The fidelity of a quantum gate is impossible to reach 100%.

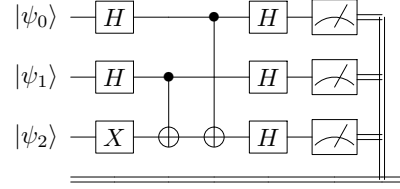


Fig. 1. An example quantum circuit. It generates a GHZ state that is commonly used to study entanglement [15].

- **Leakage.** The qubits may unintentionally be taken out of the computational subspace described by the $|0\rangle$ and $|1\rangle$ basic states, leading to leakage errors.

Quantum noise is device-specific. Different quantum computers have different noise effects. According to the construction materials and construction methods, quantum computers are classified into superconducting quantum computers, optical quantum computers, diamond quantum computers, etc. Even the same type of quantum computer may have different noise influence performance due to hardware conditions, environmental factors, and other reasons.

Although there are many noise canceling techniques proposed [16], [17], the noise cannot be eliminated completely. Thus quantum software engineering (development, testing, optimization, etc.) must take quantum noise into consideration.

D. Quantum Software Engineering

There is already a large body of work in the field of quantum software engineering [4].

Development. Many concepts and techniques in traditional software development can be applied to quantum programming [18]–[20]. Pérez-Delgado *et al.* [18] proposed an extensive set of UML (Unified Modeling Language) to efficiently construct quantum software models. Cartiere [20] presented a specification language with symbolic principles in classical software engineering to simplify the process of quantum algorithm designing and development. Many quantum programming languages [21], [22] and frameworks [23], [24] are extended from classical programming languages and frameworks.

Optimization. Due to the complexity and cost of quantum circuits, many approaches aim to optimize quantum circuits [25]–[28]. For example, Ittah *et al.* [25] proposed intermediate representations (IRs) that describe the relationship between classical and quantum data to enable dataflow optimization for quantum programs. Häner *et al.* [26] introduced an assertion-based method to show the situation of entangled subsystems for optimization. Soni *et al.* [28] explored the importance of data preprocessing and took the advantage of quantum parallelism to obtain computational speedups.

Testing. Similar to traditional software, quantum programs also need to be tested to avoid unintended behavior [29]–[35]. Miranskyy *et al.* [29] discussed some testing and debugging strategies to ensure the quality of quantum software. QuanFuzz [30] proposed a fuzz testing method that randomly generates test cases and drove the control flow into different branches

in the classical part. *Li et al.* [31] presented a projection-based assertion tool for quantum program testing. *Huang and Martonosi* [32] collected and summarized some types of bugs in quantum programs. QDiff [34] proposed a differential testing technique to detect reasonable quantum circuits that may potentially contain bugs.

Verification. There are also several approaches focused on the verification of quantum programs. *Chareton et al.* [36] introduced an automatic tool for quantum specification verification. *Feng et al.* [37] developed a CPS-like distributed programming language for quantum system verification and validation. *Ying* [38] used quantum Hoare logic, invariant generation, etc. to comprehend and verify quantum programs.

Despite a great deal of pioneering work in quantum software engineering, the impact of noise on quantum software is largely neglected, which motivates us to conduct this study.

III. OVERALL METHODOLOGY

Developers usually design quantum algorithms and programs in an ideal environment, while the actual execution environment of the developed programs inevitably contains noise. Thus, there is a gap between the expected behavior of a quantum program and its actual behavior in real quantum computers, which brings quantum software more weaknesses. Understanding the characteristics and impact of quantum noise is important for quantum software engineering. The purpose of this paper is to mend the gap by investigating the noise with real quantum computers and quantum programs, which can improve the steadiness of quantum software.

This section introduces the overall methodology of our study.

A. Quantum Computing Platforms

We mainly considered two quantum computing platforms in this paper, including:

Quantum computing cloud platform. Our study used a quantum computing cloud platform provided by a large research institute [12]. We call it *QuanCloud* for short in this paper. It offers a convenient service that allows users to access a 12-qubit superconducting quantum computer. Developers can use the visualized web interface to design quantum circuits or the QCIS instruction set [21] to program quantum circuits, and then submit them to the quantum computer for execution.

Cirq quantum simulator. Cirq [3] is a quantum simulator developed by Google. The developers can conveniently construct quantum circuits and combine them with classical programs. The simulator can produce the analog vectors of quantum circuits to denote the resulting quantum state. Cirq can also run quantum circuits directly and get the sample result from measurement gates. An important feature of Cirq that makes it preferable in our study to Qiskit and other quantum simulators is that developers can flexibly simulate average and uneven noise environments using its built-in APIs.

B. Data Collection

We also collected several real quantum programs for this study. A real quantum program is designed to solve a meaningful problem like quantum state finding (Grover), protocol

demonstration (BB84), etc. We downloaded the quantum programs possibly developed with Cirq by searching for keywords including ‘quantum’, ‘Cirq’, ‘program’, and their combinations. Perhaps for immature quantum programming, there are not many projects using Cirq. After excluding irrelevant programs (*e.g.*, projects about the development of quantum simulators, programming languages, or computation platforms), we obtained 76 code files. And then we filtered out the programs that were not executable or duplicated. For the programs that remained, we went through the following process: (1) Check whether the quantum circuit structures are supported by the quantum cloud platform. (2) If the quantum circuit uses an unsupported structure, we attempt to replace the complex quantum gate with the equivalent of appropriate gates. (3) If the structure cannot be revised or its scale after substitution is oversize, the quantum circuit will be only used in the quantum simulator or abandoned. (4) If a quantum circuit still does not work properly for some reason (running time is too long, requiring too much computing resource, etc.), it will be excluded.

Finally, we obtained 18 quantum programs that have high quality and practicality. In our study, we used these quantum programs to analyze the effects of quantum noise, test the performance of noise modeling, and demonstrate the application value of noise measurement.

C. Automated Circuit Generation

Due to the limited number of real quantum programs, we further generate plenty of quantum circuits to augment the dataset. Algorithm 1 shows the automated circuit generation.

For convenience, we use “width” and “depth” to respectively denote the number of qubits and moments in a circuit. To generate a random quantum circuit, we need to set the expected width W , the depth limit D_{limit} , and the set of available quantum gates S_g . The algorithm will return the quantum circuit c and its scale information including width W_c , depth D_c , number of gates N_g and number of multi-qubit gates N_m .

The overall process of quantum circuit generation is iteratively adding a random quantum gate to the circuit until its scale satisfies the specified size. In Algorithm 1, lines 5-8 and lines 26-28 describe how we conduct the scale parameters. If the width of the quantum circuit is smaller than the predefined limit, the algorithm will enter the recursion and return a new random circuit. However, if the depth reaches our expected value, the current quantum circuit will have a 50% chance of randomly adding quantum gates and a 50% chance of stopping adding and returning the current circuit. Once we return the current quantum circuit immediately after reaching the depth requirement, there will be only one gate at the last moment of it, which would decrease the diversity of the circuits. Lines 9-18 are about quantum gate insertion, which includes selecting a random gate type and selecting the qubit(s) to operate on. And multi-qubit gates may increase the depth of multiple qubits, which is handled in lines 16-17.

Using Algorithm 1, we can assign different combinations of W and D to generate quantum circuit sets with the normal

Algorithm 1 generate_random_circuit(W, D, S_g)

Input: The number of qubits (width), W ; The maximum number of moments (depth), D ; The set of possible quantum gates, S_g .

Output: Quantum circuit, c .

```
1:  $c = \text{circuit}()$  // Create an empty circuit
2:  $\text{depths} = [0 \text{ for } _ \text{ in range}(W)]$ 
3: while true do
4:   if  $\max(\text{depths}) \geq D_{\text{limit}}$  then
5:     if  $\text{random.random}() > 0.5$  then break
6:   end if
7:   end if
8:    $g = \text{random.pick}(S_g)$  // Pick a random gate
9:   if  $g$  is single-qubit gate then
10:     $\text{qubits} = [\text{random.randint}(1, W)]$ 
11:     $\text{depths}[\text{qubits}[0]] += 1$ 
12:   else
13:     $qb = \text{random.randint}(1, W - 1)$ 
14:     $\text{qubits} = [qb, qb + 1]$ 
15:     $d_{\text{max}} = \max(\text{depths}[qb], \text{depths}[qb + 1])$ 
16:     $\text{depths}[qb], \text{depths}[qb + 1] = d_{\text{max}} + 1$ 
17:   end if
18:    $c.\text{append}(\text{gate}=g, \text{qubits}=\text{qubits})$ 
19: end while
20:  $c.\text{append}(\text{gate}=\text{Measurement}, \text{qubits}=[Q_1, Q_2, \dots, Q_W])$ 
21: if  $W_c < W$  then
22:   return generate_random_circuit( $W, D, S_g$ )
23: end if
24: return  $c$ 
```

distribution of sizes and random inner structures, which provide sufficient data for more detailed analyses.

D. Noise Effect Measurement

Quantum state fidelity is an important metric to measure the output precision of quantum circuits in quantum information area [9]. It is computed by comparing the density matrix of a quantum state with its theoretical value. Unfortunately, the density matrix is not available in quantum cloud platforms, which can only return the probability distributions of qubits after measurement. For example, for a two-qubit system whose output is $|\psi_1\psi_2\rangle$, the result returned by a quantum computer would be a four-element vector (e.g., $[0.1, 0.2, 0.3, 0.4]$), where each element represents the probability of a basic state (e.g., the probability of $|00\rangle$ is 0.1, the probability of $|01\rangle$ is 0.2, etc.). Therefore, inspired by the Bhattacharyya Coefficient [39], [40], we use the similarity between the theoretical probability distribution of the circuit and its actual measured distribution to approximate the fidelity value, using the following formula:

$$F(P, Q) = \left(\sum_{i=1}^N \sqrt{p_i q_i} \right)^2 \quad (2)$$

where $P = (p_1, p_2, \dots, p_N)$ is the ideal probability distribution vector of a circuit and $Q = (q_1, q_2, \dots, q_N)$ is the actual result

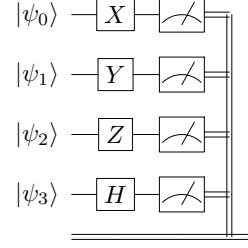


Fig. 2. The structure of the benchmark circuit.

vector measured from the computing environment.

The higher value of $F(P, Q)$ means a higher similarity between the real computing environment and the ideal, which indicates a smaller noise effect in the environment. With high-fidelity simulation, programmers can develop quantum software with high resistance to noise and determine its robustness.

IV. QUANTUM NOISE CHARACTERISTICS

This section aims to answer **RQ1: What are the characteristics of noise presented in real quantum computers?**

To answer this question, we use a simple benchmark circuit to periodically run on the real quantum computer *QuanCloud* to measure the noise. Figure 2 shows the structure of the benchmark circuit. It has four qubits, each of which passes through a single-qubit gate before measurement. The noise level can be quantified by the fidelity of its output result.

The environment of a quantum computer may change over time [41]. Quantum computers often need to stop working and perform calibration periodically to avoid error accumulation. We define a segment of continuously available time as a *calibration cycle*. Each calibration cycle of the *QuanCloud* lasts about 30 minutes.

In our experiment, we repeatedly run the benchmark circuit on the *QuanCloud* in separated calibration cycles. The shots number is 12,000 for each run, and the interval between each run is about 20 seconds. In each run, we measure the fidelity of each qubit and all qubits combined. Figure 3 shows the results obtained in two calibration cycles.

The results show the clear existence of quantum noise in *QuanCloud* (i.e., the 12-bit superconducting quantum computer). We have also observed non-neglectable amount of noise in other quantum computing platforms (e.g. Qiskit [1], IBM Q [42], Origin Quantum [43], etc.), but we can't conduct detailed analysis on them due to the limited access (i.e., insufficient permissions and/or high cost). The noise has little influence on some qubits (qubit=2 and qubit=3), whose fidelity values are generally above 0.98, while some qubits (qubit=0 and qubit=1) whose fidelity frequently goes below 0.96 with large variance are seriously affected. The main reason causing different impacts is the discrepancy in noise resistance and operation precision caused by physical construction.

Meanwhile, the fidelity of the benchmark circuit changes over time with great variance far beyond mere random effect and no clear pattern. The reasons include the unpredictability of the quantum computing environment that generates noise

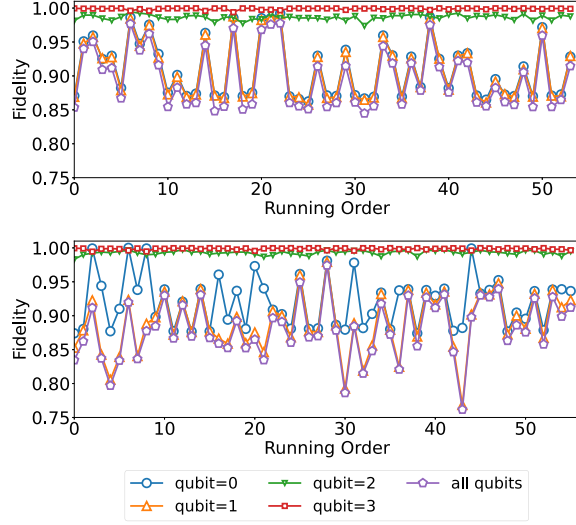


Fig. 3. The fidelity measured with the benchmark circuit on *QuantCloud* in two calibration cycles. The blue, orange, green, and red lines denote the fidelity changes over time for the four qubits with X, Y, Z, and H gates respectively. The purple lines represent the overall fidelity.

and the sampling randomness of measurement operation in quantum circuits. Such an unpredictable pattern of quantum noise makes it difficult to cancel or reduce much of its impact. However, the fidelity curves of qubit=0 and qubit=1 have the same change trend and conform to the curve on all qubits. It is possible that a single qubit affects the fidelity of a quantum circuit drastically and should be dealt with primarily. Hence, quantum software must be prepared to live with the noise.

V. PROGRAM ROBUSTNESS AGAINST NOISE

This section aims to answer **RQ2: How does the noise influence the correctness of quantum programs?**

A. Experiment Setup

This experiment is conducted by running the quantum programs under different types and levels of quantum noise.

In a quantum circuit, the effect of noise on the quantum state can be expressed as a quantum channel, which can be thought of as a special form of quantum gate [44]:

$$\epsilon(\rho) = pP\rho P + (1-p)\rho \quad (3)$$

where ρ is the density matrix (the product of the $|\psi\rangle$ and its transposition) of a quantum state, P is the noise density operator, p is the probability that the quantum state is affected by noise. The effect of noise can be seen as adding a noise channel after each quantum gate. In this experiment, we consider three channels that are recommended to simulate noise in Cirq [45]:

- **Bit-flip channel.** Bit-flip channel exchanges the α and β values of the quantum state with probability p . This quantum channel is equivalent to an X gate with probability. Its density operator is $P_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$.

- **Phase-flip channel.** The phase-flip channel acts as a Z quantum gate. Its density operator is $P_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$.
- **Depolarize channel.** The density operator of a depolarized channel will produce a noise effect whose density operator is one of the operators of X, Y, and Z gates with equal probability $p/3$.

With the help of the noise API in Cirq, we are able to simulate a variety of noise types and levels. We run the same quantum program under simulated ideal environments, simulated noise environments, and real environments to understand the impact of noise. The shot number is 1,000 for each run.

B. Robustness of Real Quantum Programs

The robustness of a real quantum program against quantum noise means its ability to retain correct behavior under the noise, which can be quantified by accuracy. Each real program has a unique oracle output, and the program will produce the correct output through statistics when the right result occurs enough times. We run each program under each noise condition 1,000 times and calculate its accuracy (*i.e.*, the number of runs that produce the correct output).

We first test the 18 Cirq-based real quantum programs using their default input parameters in different noise environments. Each noise environment uses one of the three types of noise (bit-flip, depolarize, and phase-flip) and one of 11 noise levels (*i.e.*, noise probability p in $[0, 0.1, 0.2, \dots, 1]$). For some programs, we also consider different input parameters or use different computing platforms, which will be explained later. The main findings are summarized as follows:

First, **the accuracy of most quantum programs decreases as the noise strength (*i.e.*, noise probability p) increases for all types of noise**, as shown in Figure 4. For example, the accuracy of 14 quantum programs quickly falls below 0.4 when the bit-flip noise probability reaches 0.2. However, the correlation does not always hold in all cases. The *adder* program has high accuracy with all types of noise and probability p for its special structure. The accuracy of some quantum programs will even increase when the noise probability p is too large (*e.g.*, *little_elsa* under bit-flip and *bernstein_vazirani* under phase-flip noise). That's because the circuit under the noise impact coincidentally becomes a circuit whose output state can lead to correct results in the classical part (although the internal logic may be completely wrong).

Second, **different types of noise have different impacts on the same quantum program**. Bit-flip and depolarize noise has greater damage to the correctness of most quantum programs. With the increase of p , the accuracy rate of most programs gradually approaches 0. However, phase-flip noise has little or even no effect on 9 quantum programs. The reason is that some quantum circuits in these quantum programs do not involve phase operations. The *Simple_Cirq_Program* program show completely different patterns of accuracy change under the three types of noise, because its quantum circuit has only an X gate and a measurement gate, so there is no phase computation in it and won't be affected by phase-flip noise.

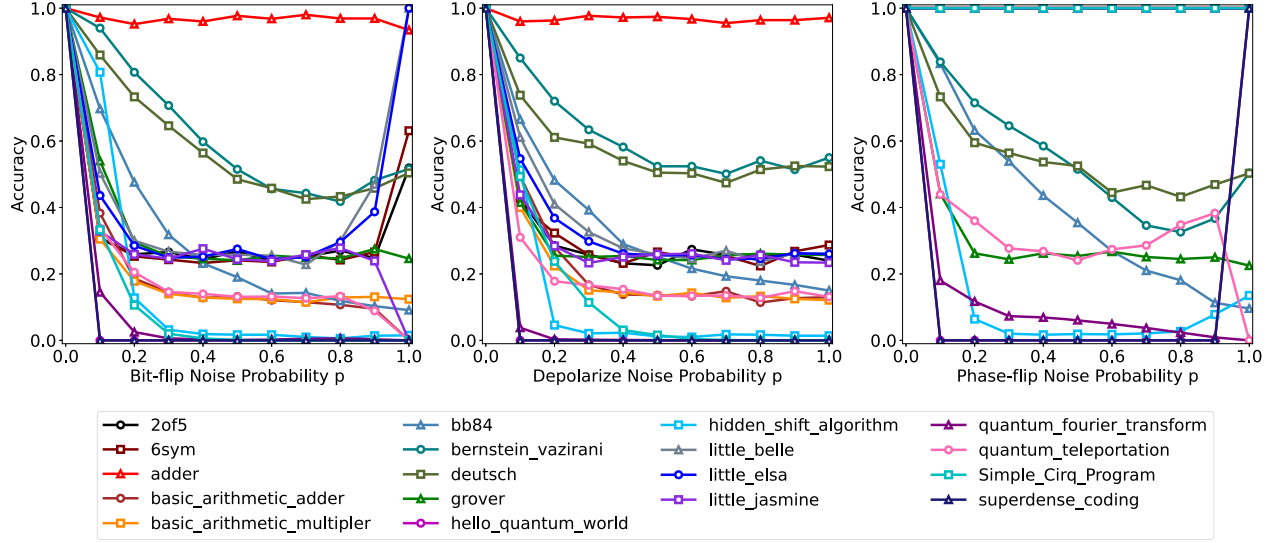


Fig. 4. Quantum program accuracy in the simulation environment. The pictures are the change curve of the accuracy rate of quantum programs with the noise probability p under the environment of bit-flip, depolarize, and phase-flip noise respectively. The notation in the legend is the name of quantum programs.

Third, **the robustness of a quantum program against noise is dependent on its size**. We define *dynamic quantum programs* as programs that may change the circuit structure based on the input parameters. For example, the *hidden_shift_algorithm* has an input parameter `qubit_count` that determines the qubit number of the circuit. We use three dynamic quantum programs including *hidden_shift_algorithm*, *bb84* and *bernstein_vazirani* to analyze the relationship between circuit scale and robustness. These dynamic circuits are composed of repeated similar structures. Figure 5 shows when noise probability p is fixed, the accuracy of these programs is inversely proportional to the number of qubits. This indicates that more qubits involved in the quantum program typically lead to greater program vulnerability against noise.

Fourth, **the accuracy of a quantum program under the presence of noise may be influenced by its input parameters**. On some quantum programs, we find that the program robustness differs with different input parameters. For instance, the *bb84* program contains three input parameters including *alice_basis*, *alice_state* and *bob_basis*. Each parameter can only be 0 or 1. Table III shows the noise impact on the program with each input parameter combination. When the three parameters are 010, 101, or 111, the noise will significantly influence the accuracy of the program, while in the other five cases it cannot. After examination, we find that it is due to the classical part of the quantum programs. For example, when the input is 000, the quantum circuit generated by the classical part only contains measurement gates. When the 1st parameter equals the 3rd parameter, the classical part will not record the results.

The same phenomenon is found on real quantum computers. We run the *deutsch* and *hidden_shift_algorithm* programs on the quantum cloud platform. Two programs have an input

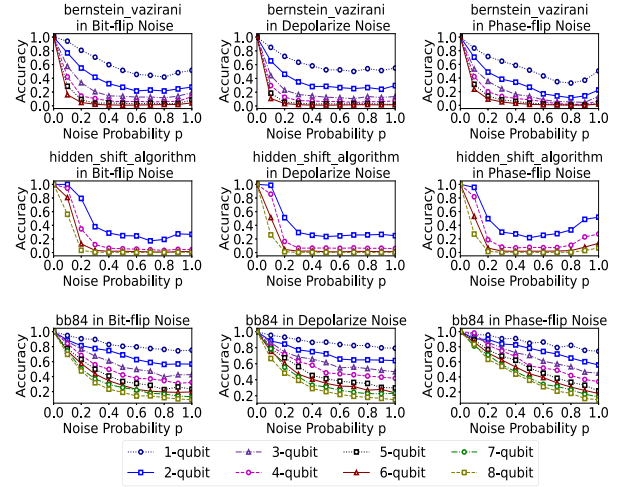


Fig. 5. The accuracy of three quantum programs with different numbers of qubits under different types and levels of quantum noise.

TABLE III
THE IMPACT OF NOISE ON *bb84* PROGRAM WITH DIFFERENT INPUT COMBINATIONS IN BIT-FLIP NOISE ENVIRONMENT AND PROBABILITY $p=0.1$.

alice_basis	0	0	0	0	1	1	1	1
alice_state	0	0	1	1	0	0	1	1
bob_basis	0	1	0	1	0	1	0	1
Accuracy	1.00	1.00	0.89	1.00	1.00	0.90	1.00	0.81

TABLE IV
THE ACCURACY OF TWO QUANTUM PROGRAMS RUNNING WITH DIFFERENT INPUTS ON THE REAL QUANTUM COMPUTER.

Program	Secret Key	Qubit	Depth	Gate	Multi-Gate	Acc.
hidden_shift	00	2	5	6	2	0.9541
hidden_shift	01	2	7	10	2	0.8921
hidden_shift	10	2	7	10	2	0.8668
hidden_shift	11	2	7	12	2	0.8942
deutsch	00	2	2	4	0	0.9862
deutsch	01	2	6	7	1	0.8352
deutsch	10	2	5	8	1	0.9010
deutsch	11	2	9	11	2	0.7966

parameter named “secret key”, which will lead to different structures of quantum circuits. We run each program with 4 possible values of the secret parameter with the shot number 12,000. Table IV shows the accuracy of each program under each input value. Program *deutsch* with secret key ‘00’ has the highest accuracy 0.9862, while the one with ‘11’ has the lowest accuracy 0.7966. This is due to the different quantum circuits generated for each input. If the circuit has higher width, depth, or number of gates, it will become less stable.

C. Analyses on Generated Quantum Circuits

We further conduct more fine-grained analyses with auto-generated quantum circuits. To explore the relationship between the quantum circuit size and its fidelity in a noise environment in more detail, we use Alg 1 randomly generate 1,200 quantum circuits, whose numbers of qubits are within the range of [3,12] and their depths are within the range of [3,14]. The number of gates in a circuit is in the range of [5, 96]. We conduct correlation analysis on the scale attributes of these circuits and their fidelity calculated after 12,000 runs on the *QuantumCloud*.

We find that **the robustness of the circuit against noise is positively correlated to the circuit size**. The attribute that has the greatest impact on the fidelity of a quantum circuit is the number of qubits (width), whose Pearson correlation coefficient is -0.6147. The Pearson coefficient of the number of gates, number of multi-qubit gates, and circuit depth are respectively -0.5597, -0.5284, and -0.2298, meaning that the increase of these three attributes will also cause the decline of the fidelity of a quantum circuit.

However, **the quantum circuits at similar scale may have significantly different fidelity**. For instance, we notice that two quantum circuits that have a similar size running in the same environment may produce very different fidelity (0.7617 vs. 0.9403). It means that a quantum circuit’s robustness against noise also depends on its structure.

To understand the relationship between the structure of a circuit and its robustness, we use the fuzz testing technique [46] to find the circuit structures that are more susceptible to quantum noise. Inspired by the *n*-gram language models [47], we define *n-gate struct* to represent a combination of *n* adjacent quantum gates on the same qubit of the quantum circuit. For instance, in a quantum circuit with 2 qubits, suppose there is X gate on qubit=1, Y gate on qubit=2, and CZ gate on these two qubits, then the *1-gate structs* of the quantum circuit include X,

TABLE V
SOME COMMON SUBSTRUCTURES AMONG THE CIRCUITS THAT ARE VULNERABLE AGAINST QUANTUM NOISE.

Struct	N	Occurrences	Struct	N	Occurrences
CZ	1	816	Y2P+SD	2	49
Z	1	794	SD+TD	2	49
X2P	1	788	Y+X2M	2	48
T	1	768	Y+Y+SD	3	6
Y2M	1	767	Y2P+X2M+T	3	6

Y, CZ, and 2-gate structs include X+CZ, Y+CZ. These *n-gate structs* are the basic sub-structures of quantum circuits.

Then we attempt to find the *n-gate structs* that are sensitive to noise (and thus reduce the robustness of the circuits). We first generate a set of vulnerable circuits whose fidelity is below a pre-defined threshold (0.3 in this paper). Each vulnerable circuit is generated by iteratively adding quantum gates into a random basic circuit. In each iteration, we add a gate to a random location in the target circuit and check whether the new circuit is vulnerable (fidelity < 0.3) to noise (depolarize channel with total probability $p = 0.1$). The process is repeated for 5 types of all gates. If none of them produces a vulnerable circuit, we select the lowest-fidelity circuit as the base circuit and go to the next iteration. After we obtain 1,000 vulnerable circuits, we extract sensitive *n-gate structs* by looking for common substructures among the circuits. Some of the sensitive *n-gate structs* are shown in Table V. We can see that **some substructures such as CZ, Z, Y2P+SD are more sensitive to the noise, meaning that the circuits that contain them may be less robust in the computing environment**.

VI. NOISE MODELING

This section aims to answer **RQ3: Whether and how can the influence of quantum noise be modeled?**

Based on analyses in Section IV, it is clear that quantum program developers have to consider quantum noise during software development. However, the impact of noise on quantum programs is complex according to Section V. Meanwhile, directly testing the programs on real quantum computers is usually expensive and even infeasible. Thus, in this section, we attempt to model the impact of noise using machine learning.

Our goal of modeling is to predict the fidelity of a quantum circuit under a certain computing environment so that the model can be used offline to estimate whether a program will behave normally in a noisy environment.

A. Dataset and Baseline

Dataset. Due to the limited access to real quantum computers, we generate the quantum circuits dataset in a Cirq-based simulation environment. To simulate the complex noise environment in real quantum computers, we use the Pauli noise model [48] whose formula is:

$$\epsilon(\rho) = \sum_j p(j) P_j \rho P_j$$

where $p(j)$ is the error rate, P_j is the Pauli operator or the identity matrix I, and ρ is the density matrix of a quantum

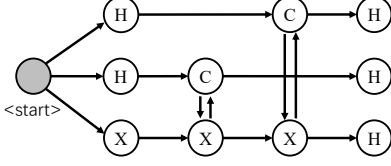


Fig. 6. The graph representation of a quantum circuit.

state. The Pauli operators include the density operators of gate X, gate Y, and gate Z, which represent three directions of real noise effect on Bloch Sphere [9]. In this noise model, each quantum state has the probability of $p(I)$ that will not change while has the probability of $p(X)$, $p(Y)$, and $p(Z)$ that will be affected by the X, Y, and Z density operators respectively.

To generate the dataset, we consider two groups of quantum gates: a group of five common gates (X, Y, Z, H, CZ) and a 13-gate extended group (X, Y, Z, H, S, SD, T, TD, X2P, X2M, Y2M, Y2P, CZ). For each group, we use Algorithm 1 whose W and D respectively are averagely assigned in the range [3, 8] and [3, 15] to generate 25,000 random quantum circuits with a similar number of circuits for each scale. In the Cirq quantum simulator, we set up three computing environments with different Pauli noise distribution e_1 , e_2 , e_3 , which are randomized while ensuring the total noise probability of $p(X)$, $p(Y)$, and $p(Z)$ equals to 0.1. Then we run each circuit 1,000 times in each of the three noise environments and compute its fidelity value as the label. In the end, we obtain six independent datasets. For each dataset, we randomly choose 80% circuits for training and the other 20% for testing.

Baseline. We implement a noise modeling method based on linear regression as the baseline for comparison. The model takes the metadata of a circuit (including width, depth, the number of gates and multi-qubit gates.) as the input feature and predicts the fidelity. The model can be represented as $y = \theta_0 + \theta_1 x_{width} + \theta_2 x_{depth} + \theta_3 x_{n_gates} + \theta_4 x_{n_multi_gates}$, and the coefficients θ_i are found by gradient descent.

B. Modeling with Graph Neural Networks

We adopt a graph neural network [49] approach to model the impact of quantum noise.

First, we explain how we convert the input quantum circuit to a directed graph that can be processed by graph neural networks. Each single-qubit gate in a quantum circuit is a node. A two-qubit gate is regarded as two nodes connected bidirectionally. An edge is added between each pair of successive quantum gates in a qubit line, directing from the former gate to the latter. To make the graph more connected, we add an empty <start> node into the graph and connect it to the first nodes of each qubit. Figure 6 shows the graph data structure of the quantum circuit in Figure 1.

Then we encode each node based on its features. In a graph neural network, node encoding mainly includes the local information of the node, which is then exchanged with neighboring nodes to obtain global information. In our model, we use the following information to represent a node:

TABLE VI
THE AVERAGE ERROR OF FIDELITY PREDICTION USING OUR GRAPH NEURAL NETWORK AS COMPARED WITH LINEAR REGRESSION.

Noise Distrib.	# Gate Types	Average Error Linear	Average Error Ours
e_1	5	0.1037	0.0511
e_2	5	0.1045	0.0496
e_3	5	0.1074	0.0536
e_1	13	0.1259	0.1006
e_2	13	0.1324	0.1037
e_3	13	0.1332	0.1026

- The type of the gate represented by the node;
- The qubit id of the gate;
- The depth of the gate in the circuit;
- The sequence of gates appeared before the gate;
- The set of n-gate structs appeared before the gate;
- The set of sensitive n-gate appeared before the gate.

Our model is based on the HGP-SL [50] architecture that achieves state-of-the-art performance on graph-level classification and regression problems. During training, we use the Adam optimizer. The number of features is 143. The learning rate is set to 1.2×10^{-4} and the weight decay is 5×10^{-4} . The training processes take about 50 epochs to converge.

C. Model Performance

The performance of our model is shown in Table VI. For each environment setting, we compute the average error of the fidelity predicted by our graph neural network model and the linear regression baseline. A lower error means that the model can estimate the impact of the quantum noise more precisely.

According to the results, in some cases, our graph neural network model outperforms the linear regression baseline. Specifically, our model achieves a low average error (around 0.05) when predicting the fidelity for 5-gate circuits, while the average error of linear regression is above 0.1. Through detailed analyses, we found that the prediction errors of our model on 90% of the quantum circuits are lower than 0.10, meaning that our model can achieve a more accurate noise impact estimation on circuits.

When the number of gate types in the circuit is increased to 13, our model still outperforms the baseline, with an average fidelity prediction error of about 0.10. However, the performance gap between our model and the baseline becomes lower (about 0.03), the reason might be that the randomly generated quantum circuits have more complex and diverse structures that increase the learning difficulty for the models.

We further examine whether our method can correctly reflect the findings we obtained through empirical analysis in Section V. We feed 36,000 newly generated quantum circuits with different sizes (widths and depths) to the model and get their fidelity prediction. We computed the proportion of low-fidelity (less than 0.5) quantum circuits in each circuit set with different combinations of widths and depths. The result is shown in Figure 7. Clearly, the fidelity predicted by our model matches our observation before (the robustness of a circuit is positively correlated to the circuit size).

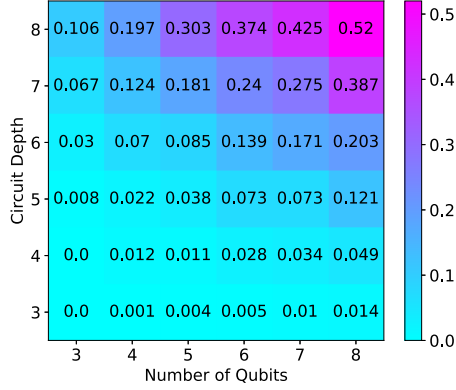


Fig. 7. The proportion of low-fidelity (less than 0.5, predicted by our model) quantum circuits in different circuit sets with the same scale.

Thus, we believe that our model can estimate the impact of quantum noise on a quantum circuit to a certain extent. Developers can use this model to measure the robustness of their quantum programs without an actual computing environment. Such ability of program robustness estimation enable various applications including noise-aware circuit design, testing, and optimization, which we leave for future work.

VII. DISCUSSION

A. Takeaways for Practitioners

Based on our study, we highlight several takeaway messages that may be useful for quantum program developers.

Quantum developers should be aware of the complexity, diversity, and dynamicity of target environments. Our study has demonstrated the general existence of quantum noise in real quantum computers and the unavoidable impact of the noise on quantum programs. Thus, when designing quantum programs, developers should consider not only the logical correctness of the program but also the influence of real noisy environments. Furthermore, the developers should understand the evolution pattern of the noisy target environment to predict the behavior of quantum programs. It is recommended to do testing to understand the feasibility and extreme scale of the programs, which will practically increase the quality of quantum software.

The quantum programs can be and should be optimized towards target environments. In Section V-C, we found that different quantum circuits at the same scale may have different fidelities under a certain environment. Even the circuits with the same function may possess different levels of robustness. For instance, the two circuits in Figure 8 are equalized in an ideal environment, but in the real quantum computer, the first circuit with S gates has a much higher probability to produce wrong results. Thus, developers can optimize their quantum programs by replacing vulnerable circuit structures (and substructures) with robust ones accordingly [51]. Using equivalent circuits with a smaller size can improve the noise resistance of quantum software.

The robustness of quantum circuits against noise can be estimated statistically. In Section VI, we used a graph neural

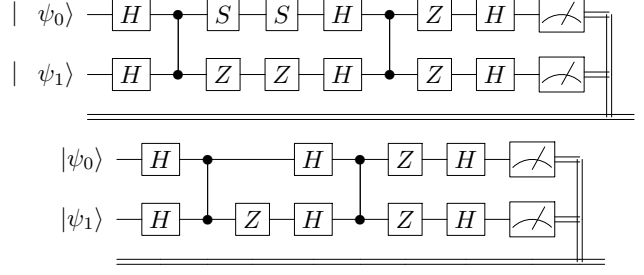


Fig. 8. Two circuit structures of the Grover algorithm (secret key='10') that are functionally equalized.

network to model the impact of certain (but unknown) quantum noise on an arbitrary circuit. The results have demonstrated the possibility of estimating these impacts based on large-scale data collected from the environment. Quantum platform and simulator developers can build more advanced models to give a more precise approximation of quantum program realistic behavior. Therefore, developers can use such models to assess the robustness of their programs with much less cost than directly testing them on real quantum computers.

B. Threats to Validity

Our findings may be biased if the quantum platform and programs that we used cannot represent the general cases. The real computing platforms that can be accessed publicly for large-scale analysis are rare, and there are not many real quantum programs due to their limited popularity. However, we have tried our best to include as many platforms and programs as possible. We also included various simulated environments and auto-generated programs to augment the dataset.

The method we used to generate circuits and simulate noise environments could be another threat, since they may not reflect real situations. To mitigate the threat, we generate plenty of random circuits with variable structures. Although there are some differences from real quantum circuits, they are all working circuits on real machines and cover most structures of real circuits. The Pauli noise channel is one of the most general models on quantum noise research [48], provided conditions close to the real environment.

VIII. CONCLUSION

In this paper, we conduct a series of studies to understand the impact of quantum noise on quantum programs. We use both the real computing environment and the simulated ones, and both the real quantum programs and auto-generated ones. We have uncovered the general existence of quantum noise, the significant impact of noise on program robustness, and the possibility of estimating noise impact with machine learning. We have also summarized several interesting findings and takeaways for quantum program developers. Our code and dataset can be found at <https://github.com/WindFrank/QuantumDataAndProgram>.

REFERENCES

- [1] "Qiskit textbook." <https://qiskit.org/textbook>. Accessed March 14, 2022.

- [2] K. M. Svore, A. Geller, M. Troyer, J. Azariah, C. E. Granade, B. Heim, V. Kliuchnikov, M. Mykhailova, A. Paz, and M. Roetteler, "Q#: Enabling scalable quantum computing and development with a high-level DSL," in *Proceedings of the Real World Domain Specific Languages Workshop, RWDSDL@CGO 2018, Vienna, Austria, February 24-24, 2018*, pp. 7:1–7:10, ACM, 2018.
- [3] "Cirq: A python library for nisq circuits." <https://balopar-cirq-rtd.readthedocs.io/en/stable>. Accessed March 14, 2022.
- [4] J. Zhao, "Quantum software engineering: Landscapes and horizons," *CoRR*, vol. abs/2007.07047, 2020.
- [5] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, 1994.
- [6] L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack," *Phys. Rev. Lett.*, vol. 79, pp. 325–328, Jul 1997.
- [7] W. van Dam, S. Hallgren, and L. Ip, "Quantum algorithms for some hidden shift problems," *SIAM J. Comput.*, vol. 36, no. 3, pp. 763–778, 2006.
- [8] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, "A quantum engineer's guide to superconducting qubits," *Applied Physics Reviews*, vol. 6, p. 021318, Jun 2019.
- [9] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge University Press, 2016.
- [10] A. Ash-Saki, M. Alam, and S. Ghosh, "Impact of noise on the resilience and the security of quantum computing," in *22nd International Symposium on Quality Electronic Design, ISQED 2021, Santa Clara, CA, USA, April 7-9, 2021*, pp. 186–191, IEEE, 2021.
- [11] A. V. Miranskyy, L. Zhang, and J. Doliskani, "On testing and debugging quantum software," *CoRR*, vol. abs/2103.09172, 2021.
- [12] "Quantum computing cloud platform." <https://www.chinadaily.com.cn/a/201710/12/WS5a0bfbf1a31061a738405c3f.html>. Accessed October 17, 2022.
- [13] D. Sofge, "A survey of quantum programming languages: History, methods, and tools," in *Second International Conference on Quantum, Nano, and Micro Technologies, ICQNM 2008, February 10-15, 2008, Sainte Luce, Martinique, French Caribbean*, pp. 66–71, IEEE Computer Society, 2008.
- [14] S. Resch and U. R. Karpuzcu, "Benchmarking quantum computers and the impact of quantum noise," *ACM Comput. Surv.*, vol. 54, no. 7, pp. 142:1–142:35, 2022.
- [15] "Greenberger-horne-zeilinger state." <https://quantiki.org/wiki/GHZ>. Accessed March 14, 2022.
- [16] C. H. Bennett and D. P. DiVincenzo, "Quantum information and computation," *nature*, vol. 404, no. 6775, pp. 247–255, 2000.
- [17] D. Bondarenko and P. Feldmann, "Quantum autoencoders to denoise quantum data," *Physical review letters*, vol. 124, no. 13, p. 130502, 2020.
- [18] C. A. Pérez-Delgado and H. G. Pérez-González, "Towards a quantum software modeling language," in *ICSE '20: 42nd International Conference on Software Engineering, Workshops, Seoul, Republic of Korea, 27 June - 19 July, 2020*, pp. 442–444, ACM, 2020.
- [19] J. Zhao, "Some size and structure metrics for quantum software," in *2nd IEEE/ACM International Workshop on Quantum Software Engineering, Q-SE@ICSE 2021, Madrid, Spain, June 1-2, 2021*, pp. 22–27, IEEE, 2021.
- [20] C. R. Cartiere, "Formal quantum software engineering: Introducing the formal methods of software engineering to quantum computing," *CoRR*, vol. abs/2111.08426, 2021.
- [21] "Pyqcisim: A python-based quantum control instruction set simulator." <https://gitee.com/quingo/pyqcisim>. Accessed March 14, 2022.
- [22] B. Ömer, "A procedural formalism for quantum computing," 1998.
- [23] M. Y. Mahmoud and A. P. Felty, "Formal meta-level analysis framework for quantum programming languages," *Electronic Notes in Theoretical Computer Science*, vol. 338, pp. 185–201, 2018. The 12th Workshop on Logical and Semantic Frameworks, with Applications (LSFA 2017).
- [24] M. Möller and M. Schalkers, "[lib]: A cross-platform programming framework for quantum-accelerated scientific computing," in *Computational Science – ICCS 2020 (V. V. Krzhizhanovskaya, G. Závodszy, M. H. Lees, J. J. Dongarra, P. M. A. Sloot, S. Brissos, and J. Teixeira, eds.)*, (Cham), pp. 451–464, Springer International Publishing, 2020.
- [25] D. Ittah, T. Häner, V. Kliuchnikov, and T. Hoeffler, "Enabling dataflow optimization for quantum programs," *CoRR*, vol. abs/2101.11030, 2021.
- [26] T. Häner, T. Hoeffler, and M. Troyer, "Assertion-based optimization of quantum programs," *Proc. ACM Program. Lang.*, vol. 4, no. OOPSLA, pp. 133:1–133:20, 2020.
- [27] P. Liu, S. Hu, M. Pistoia, C. R. Chen, and J. M. Gambetta, "Stochastic optimization of quantum programs," *Computer*, vol. 52, no. 6, pp. 58–67, 2019.
- [28] B. Soni, N. Khare, K. K. Soni, and A. Rasool, "Classical equivalent quantum based efficient data preprocessing algorithm," in *11th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2020, Kharagpur, India, July 1-3, 2020*, pp. 1–7, IEEE, 2020.
- [29] A. V. Miranskyy, L. Zhang, and J. Doliskani, "On testing and debugging quantum software," *CoRR*, vol. abs/2103.09172, 2021.
- [30] J. Wang, M. Gao, Y. Jiang, J. Lou, Y. Gao, D. Zhang, and J. Sun, "Quantfuzz: Fuzz testing of quantum program," *CoRR*, vol. abs/1810.10310, 2018.
- [31] G. Li, L. Zhou, N. Yu, Y. Ding, M. Ying, and Y. Xie, "Projection-based runtime assertions for testing and debugging quantum programs," *Proc. ACM Program. Lang.*, vol. 4, no. OOPSLA, pp. 150:1–150:29, 2020.
- [32] Y. Huang and M. Martonosi, "QDB: from quantum algorithms towards correct quantum programs," in *9th Workshop on Evaluation and Usability of Programming Languages and Tools, PLATEAU@SPLASH 2018, November 5, 2018, Boston, Massachusetts, USA (T. Barik, J. Sunshine, and S. E. Chasins, eds.)*, vol. 67 of *OASIS*, pp. 4:1–4:14, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
- [33] J. Campos and A. Souto, "Qbugs: A collection of reproducible bugs in quantum algorithms and a supporting infrastructure to enable controlled quantum software testing and debugging experiments," in *2nd IEEE/ACM International Workshop on Quantum Software Engineering, Q-SE@ICSE 2021, Madrid, Spain, June 1-2, 2021*, pp. 28–32, IEEE, 2021.
- [34] J. Wang, Q. Zhang, G. H. Xu, and M. Kim, "Qdiff: Differential testing of quantum software stacks," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 692–704, 2021.
- [35] A. V. Miranskyy and L. Zhang, "On testing quantum programs," in *Proceedings of the 41st International Conference on Software Engineering: New Ideas and Emerging Results, ICSE (NIER) 2019, Montreal, QC, Canada, May 29-31, 2019 (A. Sarma and L. Murta, eds.)*, pp. 57–60, IEEE / ACM, 2019.
- [36] C. Charetton, S. Bardin, F. Bobot, V. Perrelle, and B. Valiron, "An automated deductive verification framework for circuit-building quantum programs," in *Programming Languages and Systems - 30th European Symposium on Programming, ESOP 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings (N. Yoshida, ed.)*, vol. 12648 of *Lecture Notes in Computer Science*, pp. 148–177, Springer, 2021.
- [37] Y. Feng, S. Li, and M. Ying, "Verification of distributed quantum programs," *CoRR*, vol. abs/2104.14796, 2021.
- [38] M. Ying, "Toward automatic verification of quantum programs," *Formal Aspects Comput.*, vol. 31, no. 1, pp. 3–25, 2019.
- [39] T. Kailath, "The divergence and bhattacharyya distance measures in signal selection," *IEEE Transactions on Communication Technology*, vol. 15, no. 1, pp. 52–60, 1967.
- [40] C. A. Fuchs and C. M. Caves, "Ensemble-dependent bounds for accessible information in quantum mechanics," *Phys. Rev. Lett.*, vol. 73, pp. 3047–3050, Dec 1994.
- [41] A. S. Cacciapuoti and M. Caleffi, "Toward the quantum internet: A directional-dependent noise model for quantum signal processing," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pp. 7978–7982, IEEE, 2019.
- [42] "Ibm quantum." <https://quantum-computing.ibm.com>. Accessed March 18, 2022.
- [43] "Origin quantum cloud platform." <https://qcloud.originqc.com.cn/>. Accessed March 18, 2022.
- [44] D. V. Babukhin and W. V. Pogosov, "The effect of quantum noise on algorithmic perfect quantum state transfer on NISQ processors," *Quantum Inf. Process.*, vol. 21, no. 1, p. 7, 2022.
- [45] E. Knill, R. Laflamme, A. Ashikhmin, H. Barnum, L. Viola, and W. H. Zurek, "Introduction to quantum error correction," 2002.
- [46] H. Gascon, C. Wressnegger, F. Yamaguchi, D. Arp, and K. Rieck, "Pulsar: Stateful black-box fuzzing of proprietary network protocols," in *Security and Privacy in Communication Networks - 11th International Conference, SecureComm 2015, Dallas, TX, USA, October 26-29, 2015, Revised Selected Papers (B. M. Thuraisingham, X. Wang, and V. Yegneswaran, eds.)*, vol. 164 of *Lecture Notes of the Institute for Computer Sciences*,

- Social Informatics and Telecommunications Engineering*, pp. 330–347, Springer, 2015.
- [47] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, “A neural probabilistic language model,” *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, 2003.
 - [48] R. Harper, S. T. Flammia, and J. J. Wallman, “Efficient learning of quantum noise,” *Nat. Phys.*, no. 16, p. 1184–1188, 2020.
 - [49] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, “Graph neural networks: A review of methods and applications,” *AI Open*, vol. 1, pp. 57–81, 2020.
 - [50] Z. Zhang, J. Bu, M. Ester, J. Zhang, C. Yao, Z. Yu, and C. Wang, “Hierarchical graph pooling with structure learning,” *arXiv preprint arXiv:1911.05954*, 2019.
 - [51] J. C. Garcia-Escartin and P. Chamorro-Posada, “Equivalent quantum circuits,” 2011.